Vol. III, No. 2
November/December 1983

satn®

Software Arts Technical Notes from the creators of VisiCalc®

CONTENTS	FEATURING	PAGE
Getting Around Circular References When people repeat the adage "A place for everything and everything in its place," they're usually talking about physical objects, but it applies to programs, too. Thinking about where to put an expression on a VisiCalc sheet may be almost as important as knowing which expressions to use. Unintentional forward and circular references can create real problems, but intentional ones can be very useful. In this article we'll show you how to construct models that depend on such references to reduce the number of recalculations.	@IF @ISERROR @SUM /R	2
Using VisiCalc and ScreenWriter II Craig Stinson covers some tips on using your VisiCalc and ScreenWriter II programs together.	/PF	6
Pointer: Expanding Your @LOOKUP Range How to make your @LOOKUP range virtually as long as you want.	@LOOKUP	8
Book Review We look at another book-and-diskette package that has appeared recently, *The Executive Package*, by John Nevison. The package covers a number of business techniques and shows how to implement them using the VisiCalc program and some BASIC programs.		9
Related Products Checklist Once again we list some products that enhance the VisiCalc program.		11

GETTING AROUND CIRCULAR REFERENCES

A number of common financial analyses require you to use the final cost of a thing in calculating component costs that make up the cost of the thing itself. Interest cost is always a function of the level of debt, for example, but in some cases debt levels depend on the amount of interest. In this article we'll look at a relatively complex model, one in which six of the thirteen component costs are directly or indirectly dependent on the final cost.

The model is used to calculate the total development costs of a building and was provided to us by the DeMatteis Development Corporation, a Long Island, NY, construction firm. Certain costs depend on local fees, and for the most part we will not explain why components are calculated the way they are. We will, however, show you techniques for saving and loading circular and forward references and for reducing the number of recalculations necessary to resolve complex circular models. The original version of the model, for example, required 14 recalculations to yield the right answer. It now takes two or three, depending on the magnitude of the changes made.

Since the first problem with any model containing circular references is how to reload them after they've been developed, we'll begin at the end of the model, not at the beginning.

Successfully Loading a Circular Reference

For a brief attack on the problem, let's look at a minor variation of a problem often used to demonstrate the VisiCalc® program. The typical statement of the problem is:

SALES minus COST equals PROFIT

We've added a factor for a bonus and rewritten the problem as:

SALES minus (COST plus BONUS) equals PROFIT

That looks simple enough, but there's a catch: The bonus depends on the amount of profit. Admittedly, it's neither a conventional nor a smart way to set up a bonus, but it serves our purpose. It means that to calculate the bonus you have to calculate profit and that to calculate profit you have to calculate the bonus.

Enter the labels shown in Figure 1, and enter 400 at B2. At B3 enter:

```
>B3:.5*B2
```

Now figure the amount of the bonus. The rule we chose, purely arbitrarily, is:

BONUS equals 7% of PROFIT plus $10.00

The fixed portion ($10.00) means that the relationship between bonus and profit is not perfectly linear. It also means that it's a bit harder to rewrite the profit calculation to avoid pressing ! to recalculate the sheet. Enter:

```
>B4:.07*B5+10
```

Since B5 is currently empty, the VisiCalc program reads its value as zero, and the bonus is calculated as ten dollars.

So, to calculate PROFIT, enter:

```
>B5:+B2-(B3+B4)
```

How to Use SATN

The VisiCalc program is available on many different personal computers. These machines may have slightly different keyboards. To reduce confusion, SATN will use the symbol ® and the word RETURN synonymously to represent the keys known as ENTER, ENDLINE, and RETURN. The symbol ^ is used to represent the exponentiation operator.

To make VisiCalc documentation more readable, the program was designed so that the colon (:) has the same effect as RETURN in some commands, as when ending the > command and to separate the various elements of the replicate command.

In examples of what to type, type exactly what you see and press RETURN at the end of every line. On some computers, you may find it necessary to add parentheses to some of the expressions included in the text. For example, 3+4*5 may become (3+4)*5.

Throughout SATN, one typeface represents both VisiCalc input and output (what you should type and what you will see on the screen). The typeface looks like this:

```
This line is typeset in OCR-B.
So is this: +A1+B1*3.25
```

Characters you should type in are printed in rust.

To allow you to type the formulas directly into VisiCalc, the examples use specific coordinates, like A1, rather than the names of the value they represent. When actually typing a formula, you can use the arrows to point rather than using the > command.

SATN is published bimonthly by Software Arts Products Corp., 27 Mica Lane, Wellesley, MA 02181. Address all editorial correspondence to the Editorial Department, SATN, P.O. Box 99, Newton, MA 02162. Requests for special permission or bulk orders should be addressed to the publisher.

Domestic subscriptions: 6 issues $30.00, 12 issues $50.00; outside U.S.A., 6 issues $50.00, 12 issues $90.00. Single-copy price, $6.00. Back-issue price, $7.00. Send subscriptions, change of address, and fulfillment questions to SATN Subscriptions, P.O. Box 97, Newton, MA 02162.

Copyright © 1983, Software Arts Products Corp. All rights reserved. This periodical may not be copied or reproduced in any fashion either in whole or in part without the prior written consent of Software Arts Products Corp.

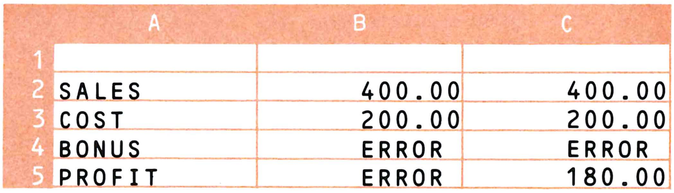
Software Arts Products Corp. makes no warranty, express or implied, with respect to the quality, accuracy, or freedom from error of the information contained in SATN, the journal for VisiCalc users, and specifically disclaims all liability for any damages resulting from the use of such information.

Unless specifically stated otherwise in any article in this issue of SATN, Software Arts Products Corp. has not examined any products and/or publications identified in SATN to determine their accuracy and/or ability to perform as advertised. Mention of a product and/or publication in SATN does not constitute an endorsement by Software Arts Products Corp. of any such product or publication.

Software Arts is a trademark of Software Arts, Inc. and Software Arts Products Corp. Unless specifically stated otherwise, all references in this issue of SATN to Software Arts shall refer to both companies. DIF is a trademark of Software Arts Products Corp. Keystroke Memory is a trademark of Software Arts Products Corp. SATN is a registered trademark of Software Arts Products Corp. VisiCalc is a registered trademark of VisiCorp.

Apple is a registered trademark of Apple Computer, Inc. IBM is a registered trademark of International Business Machines, Inc. Atari is a trademark of Atari, Inc. Commodore is a trademark of Commodore International. HP-83 and HP-85 are trademarks of Hewlett-Packard Company. TRS-80 is a trademark of Radio Shack division of Tandy Corp.

Entering that expression will give you 176.70 as PROFIT and will drive BONUS to 23.30, both of which are incorrect. Press the recalculation key (!) a few times, until the values at B4 and B5 stop changing. They will settle on 22.43 and 177.57, respectively. The expression at B4 is a forward reference: In order to resolve, it must have information that is not yet available. The expression at B5 creates a circular reference: In order to resolve, it needs information at B4, which depends on B5, which depends on B4, ad infinitum. But, as you've seen, repeated recalculations ultimately yield the correct answer.



	A	B	C
1			
2	SALES	400.00	400.00
3	COST	200.00	200.00
4	BONUS	ERROR	ERROR
5	PROFIT	ERROR	180.00

FIGURE 1. *This figure shows the sheet, formatted with /GF$, illustrating how to load circular references. In column B the expressions on rows 4 and 5 are: >B4:.07∗B5 + 10 and >B5: + B2-(B3 + B4). In column C we used an @IF statement containing the @ISERROR function as described in the text. Upon loading, the expressions in column B lock into an unbreakable ERROR condition, while those in column C will yield an answer on recalculation.*

There's yet another problem with the profit calculation. When you load a saved sheet, the VisiCalc program sets all calculated values on the sheet to ERROR and then performs a recalculation before displaying the sheet. When that first recalculation is performed at B4, the program reads ERROR at B5 and so evaluates B4 as ERROR. When B5 is calculated, it reads the ERROR just evaluated at B4, and it too evaluates to ERROR. It is caught in an endless loop, which you must somehow break. That's where @IF and @ISERROR come in.

Rather than edit B5, replicate B2...B4 into C2...C4. Enter:

```
>B2:/R...B4:C2:RR
```

Now enter the following @IF statement:

```
>C5:@IF(@ISERROR(C5),C2*.45,
                  C2-(C3+C4))
```

The key thing to understand about that expression is that it refers to itself: It is self-referencing. The expression >C6:@IF(@ISERROR(C5),C2∗.45,C2–(C3 + C4)) would perform the same test and calculation as the one at C5, but behave entirely differently, because all the references at C6 would be to cells at other locations: It is not self-referencing. Self-referencing cells, like circular references, create loops that are difficult to load, though most of them can be broken with @ISERROR.

To see how differently the expressions at C5 and B5 behave, save the sheet you have created and then reload it. Enter:

```
/SSQUICK
```

When the sheet has been saved, enter:

```
/SLQUICK
```

Now notice that the expressions at B4, B5, and C4 read ERROR, but the one at C5 reads 180.00. The expressions at B4 and B5 are in a loop (B4 reads ERROR at B5, which reads ERROR at B4, which reads ERROR at B5) that cannot be broken without blanking one of them and reentering it. At C4 and C5, though, there is no such lock. When C4 reads ERROR at C5, it evaluates to ERROR. But when C5 reads ERROR at C5 it cannot evaluate to ERROR because of the @ISERROR statement, so it evaluates C2∗.45 (an approximation of the ratio of C2 to C5, which we arrived at by trial and error). On recalculation, because C5 is no longer ERROR, the expression evaluates C2–(C3 + C4).

In such simple cases you only have to press the recalculation key (!) a few times to get the right answer. Or, using algebra, you could recast the expression as .07∗(B2–B3) + 10/1.07, and not have to recalculate at all, though it hardly seems worth rewriting the expression merely to avoid one or two recalculations. But when your model is much larger, containing many more interrelated expressions, rewriting each of them can be a tedious chore. What's more, you may never be able to save enough recalculation time to repay a two or three day investment in rewriting an entire model.

The DeMatteis Model

One of the approaches used by DeMatteis Development Corporation for estimating "TOTAL DEVELOPMENT COST" (TDC) contains a number of interrelated calculations that depend on TDC. For example, the DEVELOPER'S FEE, which must be included in TDC, is defined as 1% of TDC. In the labels shown in Figure 3 those calculations are identified by an asterisk preceding their names.

One problem with models containing a large number of necessary forward references is that they often require numerous recalculations before they converge on a correct answer. In such cases, one way around repeatedly pressing ! and waiting until the numbers stop changing is to set up several columns that use the results of each preceding column. The first column contains all the basic calculations you need to get the answer you've designed the model to give you. Immediately to its right is a second column that uses the result of the first. Since the first column produces a result, the result in the second column is closer to the final one than the first. A third column, then, would draw closer to the final answer; a fourth closer still. Exactly how many columns are required will depend on the model being resolved, but it will usually be substantially fewer columns than the number of recalculations required.

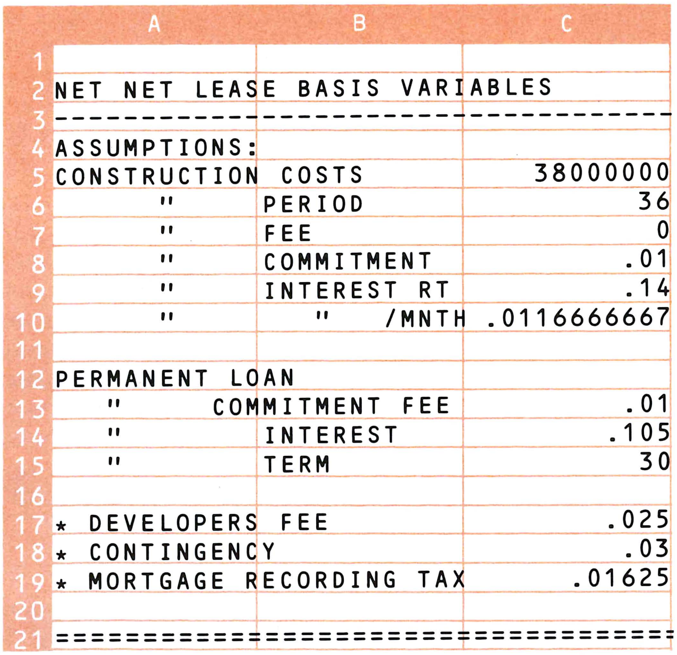
To see in detail how this works, set up the DeMatteis model.

Beginning with a clear sheet, enter the global commands:

```
/GRM
/GFI
/GC12
```

Now enter the labels and values shown in Figure 2. These values constitute the major variables on the sheet and are placed in the upper left corner. Notice that all of the values in the figure are locally formatted with /FG. There is one calculation among them, which converts years to months. Enter:

```
>C10:/FG+C9/12
```



```
         A              B              C
 1
 2 NET NET LEASE BASIS VARIABLES
 3 -----------------------------------------
 4 ASSUMPTIONS:
 5 CONSTRUCTION COSTS               38000000
 6       "        PERIOD                  36
 7       "        FEE                      0
 8       "        COMMITMENT             .01
 9       "        INTEREST RT            .14
10       "           "     /MNTH .0116666667
11
12 PERMANENT LOAN
13    "      COMMITMENT FEE              .01
14    "          INTEREST               .105
15    "          TERM                     30
16
17 * DEVELOPERS FEE                     .025
18 * CONTINGENCY                         .03
19 * MORTGAGE RECORDING TAX          .01625
20
21 =========================================
```

FIGURE 2. *Notice that the values shown on this part of the sheet are not formatted to the default, but with /FG so that they will show fuller precision. The three labels that begin with asterisks identify the factors that create circularity in the model.*

Figure 3 shows the beginning of the main part of the model. Enter the labels shown in the figure. The first entry in the model is construction costs, which is brought down from above with the entry:

```
>C23:+C5
```

Since we'll be using at least one more column, replicate that entry into column D. Enter:

```
>C23:/R:D23:N
```

The values at C24...C28 are all entered values, not expressions, so enter them as shown in Figure 3. Replicate them into column D with the entry:

```
>C24:/R...C28:D24
```

The first calculated value in this area of the model is at C29 and is a straightforward multiplication. As you enter the next series of calculations you will replicate them into column D to produce the first two columns. Enter and replicate:

```
>C29:+C5*C8®/R:D29:NN
```

The next three expressions all use TOTAL DEVELOPMENT COSTS directly. That means that they can be replicated into column D without change, since, as we mentioned above, identical expressions may behave differently when they are in different locations. At C30 the expression +C18*C37 is a forward reference; at D30 it is not (assuming you have not changed the order of calculation to row). Enter and replicate:

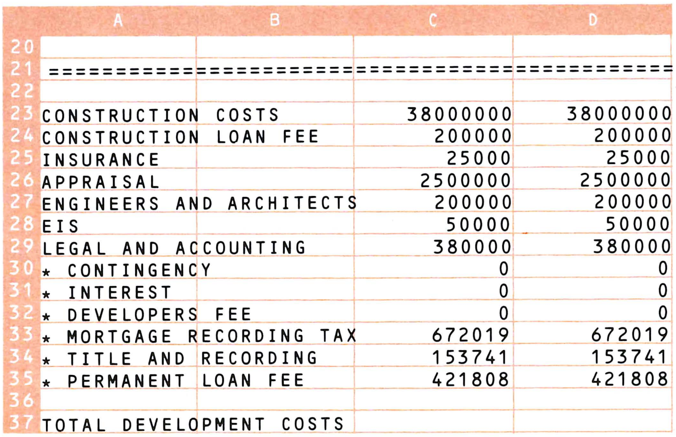
```
>C30:+C18*C37®/R:D30:NN
>C31:+C37/2*C10*C6®/R:D31:NNN
>C32:+C17*C37®/R:D32:NN
```

Since there's nothing entered at C37 yet, the expressions at C30...C32 will all read 0 (the VisiCalc program evaluates blank cells and labels as zeroes), as shown in Figure 3.

The next three expressions use TOTAL DEVELOPMENT COSTS indirectly, by including the expressions at C30...C32 within @SUM ranges. Notice, as you replicate these three expressions, that the value references are not all NO CHANGE. After we've entered the totals for the columns we'll explain why. Enter and replicate:

```
>C33:+C19*@SUM(C23...
                      C31)®/R:D33:NRR
>C34:41000+(.001*@SUM(C23...C33,
                      -15000000))*2.26
>C34:/R:D34:RR
>C35:+C13*@SUM(C23...
                      C34)®/R:D35:NRR
```

Your sheet should now look like the one shown in Figure 3.



```
         A             B               C           D
20
21 ==================================================
22
23 CONSTRUCTION COSTS            38000000    38000000
24 CONSTRUCTION LOAN FEE           200000      200000
25 INSURANCE                        25000       25000
26 APPRAISAL                      2500000     2500000
27 ENGINEERS AND ARCHITECTS        200000      200000
28 EIS                              50000       50000
29 LEGAL AND ACCOUNTING            380000      380000
30 * CONTINGENCY                        0           0
31 * INTEREST                           0           0
32 * DEVELOPERS FEE                     0           0
33 * MORTGAGE RECORDING TAX        672019      672019
34 * TITLE AND RECORDING           153741      153741
35 * PERMANENT LOAN FEE            421808      421808
36
37 TOTAL DEVELOPMENT COSTS
```

FIGURE 3. *This figure shows the sheet before adding the @IF statement.*

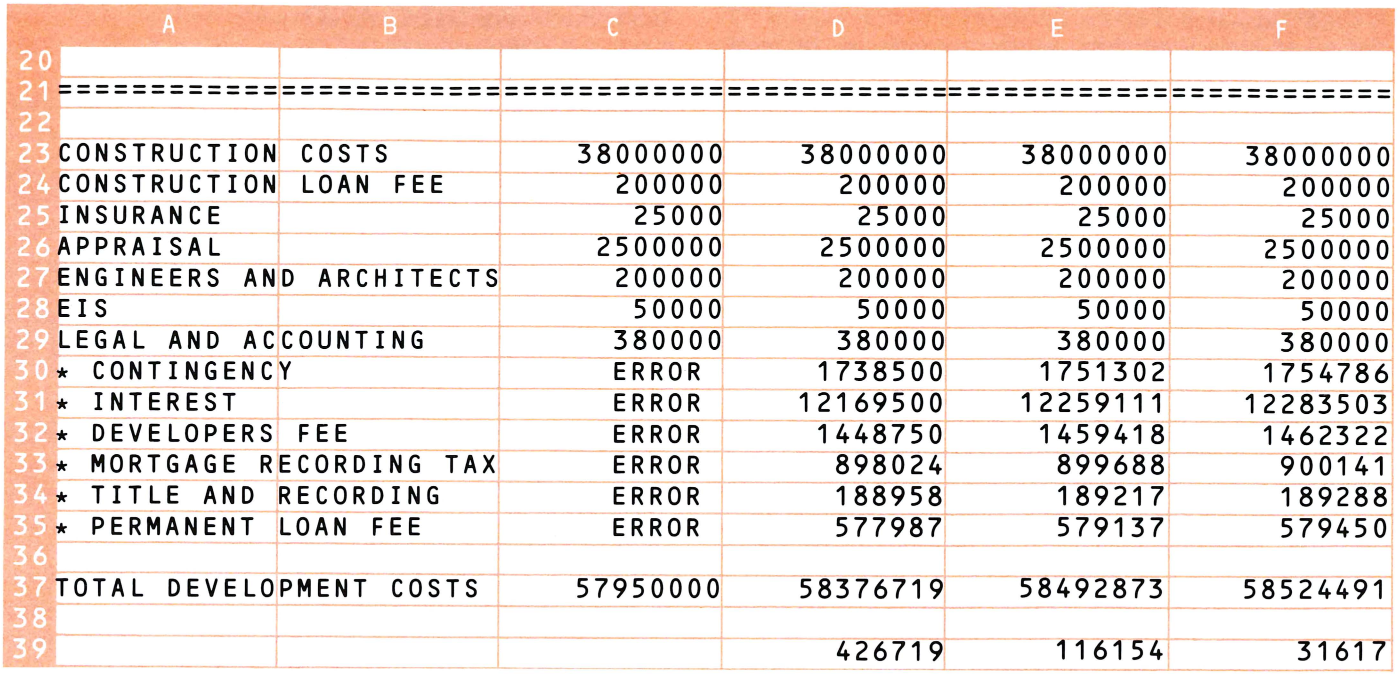
To total the column and make it possible to load the sheet, use an expression based on the @IF(@ISERROR...) statement developed earlier. Enter:

```
>C37:@IF(@ISERROR(C37),C5,
                   @SUM(C23...C35))
```

You will get 42602567 when you enter that expression. If you were to save and reload the sheet now, you would get 38000000, the value at C5.

Since there are no forward references in column D, an unadorned @SUM statement is sufficient. Enter and recalculate:

```
>D37:@SUM(D23...D35)®!
```



	A	B	C	D	E	F
20						
21	================	================	================	================	================	================
22						
23	CONSTRUCTION	COSTS	38000000	38000000	38000000	38000000
24	CONSTRUCTION	LOAN FEE	200000	200000	200000	200000
25	INSURANCE		25000	25000	25000	25000
26	APPRAISAL		2500000	2500000	2500000	2500000
27	ENGINEERS AN	D ARCHITECTS	200000	200000	200000	200000
28	EIS		50000	50000	50000	50000
29	LEGAL AND AC	COUNTING	380000	380000	380000	380000
30	* CONTINGENC	Y	ERROR	1738500	1751302	1754786
31	* INTEREST		ERROR	12169500	12259111	12283503
32	* DEVELOPERS	FEE	ERROR	1448750	1459418	1462322
33	* MORTGAGE R	ECORDING TAX	ERROR	898024	899688	900141
34	* TITLE AND	RECORDING	ERROR	188958	189217	189288
35	* PERMANENT	LOAN FEE	ERROR	577987	579137	579450
36						
37	TOTAL DEVELO	PMENT COSTS	57950000	58376719	58492873	58524491
38						
39				426719	116154	31617

FIGURE 4. *This figure shows the sheet as it appears just after it is loaded. The value at C37 is 1.525 times the value at C5. That number is then fed into the calculations in column D.*

Take a moment to examine the two columns on your sheet. The values in D33...D35 are substantially higher than those in C33...C35. Each of those series of expressions refers to the calculations above, on rows 30...32, both of which are identical and both of which refer to C37. But when cells D30...D32 are calculated, the value at C37 has been calculated. When C30...C32 are calculated, the value at C37 has not yet been calculated. When you replicated the expressions at C30...C32 into column D without any changes you converted them from forward references into conventional references. Once those expressions have been converted into conventional references, any expressions that refer to them can be calculated correctly at once (assuming no new forward references are introduced). Thus, when you replicated C33...C35 into D33...D35, you were able to make the value references relative.

Now, to check how much of a difference there is between the two columns, enter:

```
>D39:+D37-C37
```

When the model is entirely resolved, the expression at D39 will go to 0. If you were to press the exclamation point repeatedly, you could resolve the model. As you did so, you'd notice that at each recalculation, the value at C37 would be the same as D37 had been on the preceding recalculation. But to eliminate those recalculations, we can add a number of columns to the right. They will lengthen the recalculation time, but still not take as much time as a series of individual recalculations.

The question is how many columns should we add? Since it took 14 recalculations to resolve the model when it was a single column, we originally added one for each recalculation required and then threw in two more for good measure. That took us to column S, but when we reloaded the sheet and examined the expressions on row 39, it became clear that we needed only about half as many columns as we used. We kept editing down and reloading the sheet, finally settling on eight columns to the right of column C, which means column K. So, now replicate column D into columns E...K. Enter:

```
>D23:/R...D32:E23...K23:NNNNRRNNNR
>D33:/R...D34:E33...K33:NRRRR
>D35:/R...D39:E35...K35:NRRRRRR
```

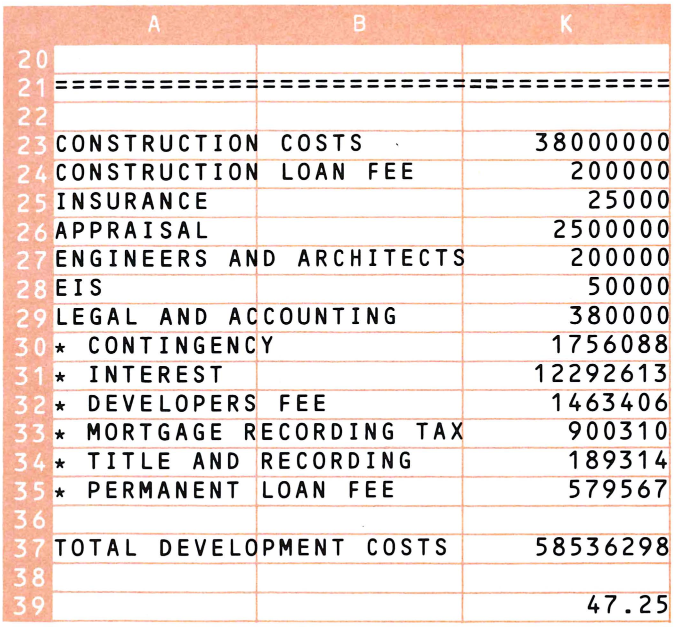
Now each column feeds its result into the column to the right and draws closer to the actual TOTAL DEVELOPMENT COSTS. We can get even closer, however, if we feed the TOTAL DEVELOPMENT COSTS calculated at column K into column C. Column C is a starting point. If we take the last calculation, at column K, and use it as the first when we repeat the calculations, we can get close to the final result more quickly. So go back to C37 and edit it or enter:

```
>C37:@IF(@ISERROR(C37),1.525*
                    C5,@SUM(K23...K35))
```

As above, we used 1.525*C5 as an approximation of the final result based on our experience with the relationship between construction costs and TOTAL DEVELOPMENT COSTS. Each model will differ.

Wrapping Up

Not all models lend themselves perfectly to the techniques shown in this article. In fact, future versions of the VisiCalc program are not guaranteed to set the entire sheet to ERROR on load, which may make the use of an @IF(@ISERROR...) statement unnecessary. But if you have some models that take several recalculations to solve, it may pay to look them over to see whether the techniques shown here could save you time. Finally, we should point out that merely changing the order of recalculation from /GOC to /GOR often eliminates a number of recalculations. Sometimes a single cell in the wrong place can create such a problem. To find such cells change the recalculation order, resave the sheet with a new name, and reload it. If ERRORs still appear, look for the cell nearest to A1 (that is, the uppermost left corner) that displays ERROR. Relocating or editing that cell can often reduce the number of recalculations the sheet requires.



```
            A          B          K
20
21 ====================================
22
23 CONSTRUCTION COSTS          38000000
24 CONSTRUCTION LOAN FEE         200000
25 INSURANCE                      25000
26 APPRAISAL                    2500000
27 ENGINEERS AND ARCHITECTS      200000
28 EIS                            50000
29 LEGAL AND ACCOUNTING          380000
30 * CONTINGENCY                1756088
31 * INTEREST                  12292613
32 * DEVELOPERS FEE             1463406
33 * MORTGAGE RECORDING TAX      900310
34 * TITLE AND RECORDING         189314
35 * PERMANENT LOAN FEE          579567
36
37 TOTAL DEVELOPMENT COSTS     58536298
38
39                                47.25
```

FIGURE 5. *By setting a title while the cursor is in column B, you can then scroll columns C-J out of sight before saving the file. When the sheet is reloaded, only columns A, B, and K will show, as in this figure. One recalculation will drive the check value at K39 to 0.00 (or .0014 if you use /FG).*

USING VISICALC AND SCREENWRITER II

by Craig Stinson

ScreenWriter II, a word processor using the Apple II's high-resolution graphics capability, generates a 70-column display with upper- and lowercase letters. VisiCalc® users will find ScreenWriter II handy for editing any VisiCalc files that don't extend beyond 70 columns. You won't have to make mental adjustments for the wraparound on screen of lines that go past the 40 column limit of the Apple in text mode.

Unfortunately, ScreenWriter II doesn't scroll horizontally. That means it won't send more than 70 characters on a line to your printer, even if your printer is in a condensed mode and can handle 132-character lines. So if you wish to incorporate a wide VisiCalc print file into a ScreenWriter II text document, you'll need first to break up your VisiCalc file into sections no wider than 70 columns. That's best done in the VisiCalc program before you make your print file.

If your VisiCalc sheet has labels to the left of, say, 18 or 24 columns of numbers, you may want to copy those labels so they appear before each section of the table as it will be printed by ScreenWriter II. That copy procedure is also better handled from within the VisiCalc program. ScreenWriter II has facilities for copying, moving, or deleting blocks of text, but it doesn't have column-block capability.

To include your VisiCalc file in a ScreenWriter II document you have some choices. If you want to be able to edit your VisiCalc material within your ScreenWriter II document, you can use ScreenWriter II's merge command: M[file name],[Sn,Dm], where Sn and Dm represent optional slot and drive designators. The command, which must be issued from the ScreenWriter II editor's command line, not from insert or change mode, causes the VisiCalc file to be inserted at the current location of your cursor. Text ahead of the cursor is pushed forward to make room for the incoming document.

If you have no particular reason to put VisiCalc material and ScreenWriter II material into a single file (but

BACK ISSUES AVAILABLE

Back issues of SATN are now available. The cost is $7.00 per copy for orders mailed within the USA. If your order is to be mailed outside the USA, add $2.50 for shipping and handling to the full price of the order.

To order back issues, please send your name and address, a list of the specific issues of SATN you want (by Volume and Issue numbers), and a check or money order payable to SATN to:

SATN Back Issues
P.O. Box 99
Newton Lower Falls, MA 02162

wish to have them assembled at output time), you can append the VisiCalc file rather than merge it.

Some ScreenWriter II commands are embedded in the document but not executed until the document is printed by the runoff program. An embedded command is usually a period followed by two letters and a name or number argument. The command must appear on a line by itself. The embedded command .AF [file name] causes ScreenWriter II's runoff program to pick up the named file and add it to whatever has been printed. Like all of ScreenWriter II's other embedded commands, the append-file command must be entered on the left side of the screen (starting in column 1) and must be followed by a carriage return.

When using the append procedure, you can drop a VisiCalc table into an already completed word processing document. If you have two or more disk drives, you can use ScreenWriter II's save-segment command to create subfiles of your text document. You'll need first to enter your subfile segment into ScreenWriter II's Get Buffer, then issue the appropriate save command.

If you want to put a VisiCalc table into an already written ScreenWriter II document by means of append commands, but you don't have two disk drives, you won't be able to save segments. The procedure is simple nevertheless. Call up your text document, move your cursor to the point where you wish to insert the VisiCalc material, enter the append-file command, followed by the name of the VisiCalc file, then delete the rest of your document and save. You'll be saving under a new file name, naturally; ScreenWriter II virtually requires that you do this, even for ordinary editing. Then get your VisiCalc figure into the ScreenWriter II editor and put another append file command at the end of it. Finally reload your original ScreenWriter II input file and delete everything ahead of the place where your VisiCalc material is to appear.

A third way to incorporate VisiCalc figures into text is to take advantage of ScreenWriter II's print chaining features. That is, you can go to the Text File Page of the ScreenWriter II's runoff program and list your VisiCalc print files in the appropriate sequence with your ScreenWriter II text files. This is essentially a way to issue several print commands at one time, allowing you to put your printer in a batch mode.

There's a difference between the batch mode and the append procedure. When in batch mode, page heading and numbering parameters are reset for each new file printed. The append command causes the program to treat each new file printed as a part of the same large document.

When you're appending or merging VisiCalc material, you may want to take advantage of ScreenWriter II's pagination commands. These will ensure that your tables are not disturbed by untimely page breaks.

The command .NP starts a new page; the command .TP n (where n is a number) starts a new page only if there are fewer than n lines remaining on the current page. The second of these commands, the conditional page break, is especially handy if there's any possibility you'll be editing the material positioned above your VisiCalc table. The immediate page break command is more likely to serve your needs within the VisiCalc portion of your document; it allows you to decide how to divide your tables when they won't fit entirely on a single page.

ScreenWriter II has commands to suppress and reinstate page numbers and headers which may be useful if numbers and headers detract from the readability of your VisiCalc figures. Or you may want to change the header (either its content or its position) for the VisiCalc portion of your document. ScreenWriter II allows you to set a new header anywhere within a document, or to specify as many as four headers at the beginning of the document and switch between them. Headers, incidentally, may as easily be footers: You specify the vertical and horizontal position of your titles in the same command in which you specify their text.

If you suppress page numbering temporarily (by means of the command .NN), ScreenWriter II will continue to count pages internally, so that when you reinstate numbering (by issuing a .NU), the pages will still be numbered correctly. You can override the program's page count if you wish. You may also switch between Roman and Arabic numerals for your page numbers.

When your VisiCalc file is narrower than your text and you wish to center it on the page, you'll probably want to change your left and right margins before and after the table. ScreenWriter II allows you to reset margins when and as you wish, by means of embedded commands. What you see on screen will not be exactly what you get at print time (all lines on screen will still start at column 1), but it should be easy enough to calculate the margin settings appropriate for creating your table. Multiply the number of columns in your VisiCalc print file by the column width to get the width of your table. ScreenWriter's default margin settings are 10 and 70, leaving a page width of sixty columns. Subtract the width of your VisiCalc table from 60 (or whatever number is appropriate if you've overridden the ScreenWriter II defaults) and divide the remainder by two; add that number to the number for the left margin and subtract it from the right to get your new margin settings.

Be aware that a decimal point in column 1 will be misinterpreted by ScreenWriter II as the start of an embedded command. VisiCalc itself won't put decimal points in column 1 (unless they begin a label). However, if you edit your VisiCalc print file, either leave those leading blanks in, or make sure you have at least one space to the left of a decimal point.

Craig Stinson is the editor of Softalk for the IBM PC.

POINTER: EXPANDING YOUR @LOOKUP RANGE

At first glance, it appears that any @LOOKUP range is limited to 254 values, since there are 254 rows and 63 columns on a VisiCalc® sheet. Usually this is quite sufficient, but occasionally you might want a longer range. It's easy to do:

We'll start the @LOOKUP range in column A (the values to be returned will be entered in column B), and we'll continue the range in column C, with the corresponding values entered in column D. Using two columns for the @LOOKUP range means that we can include about 500 values.

To illustrate the technique, we'll use integers as the index values and multiply each index value by 10 to arrive at its corresponding @LOOKUP value.

First, set recalculation to manual:

```
/GRM
```

We'll enter the index value in G1; the corresponding @LOOKUP value will be returned in G2.

Enter the labels flush right in F1 and F2 as shown in Figure 6 (remember to start the entry at F2 with a quotation mark). Now enter:

```
>G1:502
>G2:@LOOKUP(G1,A1...A254)
```

The value at G2 is 0 and will remain so until we recalculate the sheet.

The first 253 rows will be set up as usual for an @LOOKUP table. Enter the following:

```
>A1:1
>B1:10*A1
>A2:1+A1
>B2:10*A2
>A2:/R...B2:A3...A253:RR
```

That sets up a 253-row @LOOKUP range. In row 254 we'll create the bridge that will continue the @LOOKUP range into column C. In A254, you would enter a value greater than the previous value but less than the next in the @LOOKUP range. Enter:

```
>A254:253.1
```

In B254, we include the @LOOKUP function that will direct the VisiCalc program to column C:

```
>B254:@LOOKUP(G1,C1...C254)
```

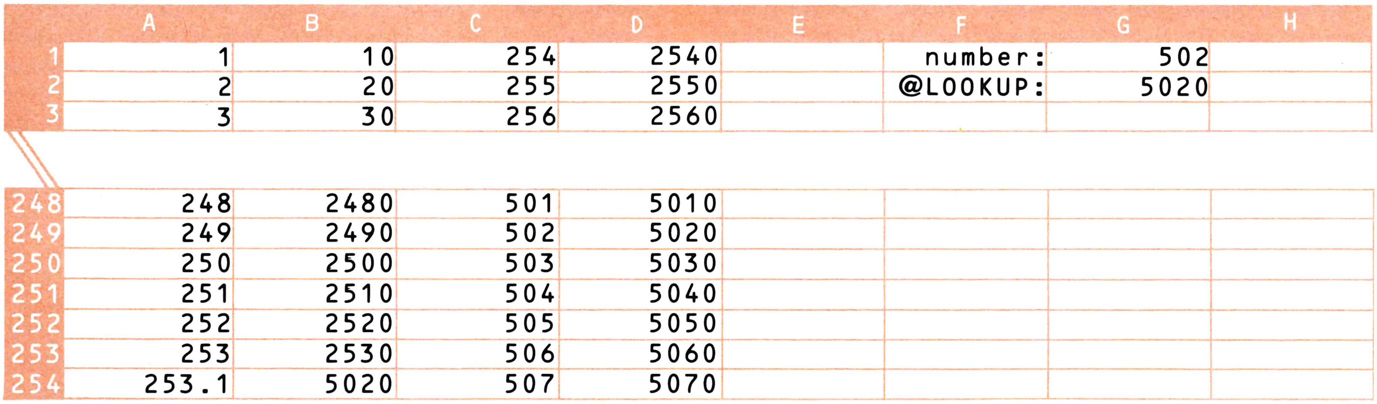
What we've done in effect is to embed an @LOOKUP function in the original @LOOKUP function in G2. When the VisiCalc program goes to evaluate G2, it looks up the value at G1 in column A. If it can't find the value, it lands on B254, which directs it to look up the value in column C. The proper value is returned at B254 and, when the sheet is recalculated, returned at G2.

Now simply enter the rest of the @LOOKUP values:

```
>C1:254
>D1:10*C1
>C2:1+C1
>D2:10*C2
>C2:/R...D2:C3...C254:RR
```

The @LOOKUP range is now 253 + 254, or 507, rows long.

The @LOOKUP range can be extended down additional columns as well, but be careful: For each additional column you will have to do an additional recalculation to return the correct number if the index value is in the last column; each recalculation brings the correct value to the bottom of the previous @LOOKUP column. For example, with three @LOOKUP columns, you will have to do two recalculations to return the correct value if the index number is in the third column.



	A	B	C	D	E	F	G	H
1	1	10	254	2540		number:	502	
2	2	20	255	2550		@LOOKUP:	5020	
3	3	30	256	2560				
248	248	2480	501	5010				
249	249	2490	502	5020				
250	250	2500	503	5030				
251	251	2510	504	5040				
252	252	2520	505	5050				
253	253	2530	506	5060				
254	253.1	5020	507	5070				

FIGURE 6. *This sheet contains an @LOOKUP table with 507 entries. The index values are in columns A and C; the values returned are in columns B and D. When the @LOOKUP function at G2 is evaluated, it searches down column A. If it doesn't find the corresponding value, it lands on cell B254, which instructs it to continue looking in column C.*

BOOK REVIEW

Reviews published in SATN are the opinions of the reviewers, not of Software Arts, and may not be construed as official positions or endorsements by Software Arts.

The Executive Package–Computing for the Corporate Manager with VisiCalc and BASIC

by John M. Nevison. Alpha Software Corp., Burlington, MA, 380 pp., $145. (Available for the IBM Personal Computer and compatibles and for the Wang Professional Computer.)

Reviewed by Robert Hildebrand

In 1980 John Nevison wrote *Executive Computing*, whose premise was that microcomputers had an important place in business. *Executive Computing* tackled business problems with programs written in the BASIC programming language and was directed at businesspeople who knew little about computers. Included was a three-page appendix about a newly released "computer product" called VisiCalc®.

Within three years, Nevison's premise was accepted universally and the VisiCalc program became a major force in restructuring professional life around microcomputers. Now Nevison and his colleagues have updated the earlier work and produced *The Executive Package—Computing for the Corporate Manager with VisiCalc and BASIC*, published by Alpha Software.

The package (book, diskettes, and audio cassette) carries through the original premise, presenting a series of business problems that face a fictitious company and showing how they can be analyzed using a microcomputer. The problems range from pricing and managing cash flow through decision analysis and forecasting. It is an ambitious undertaking in that it attempts to create an entire business world and show how a computer can fit into it.

The package has mixed success for VisiCalc users. They are unlikely to learn very much about the VisiCalc program, but they are likely to learn some valuable information about BASIC, which is sometimes a **more appropriate business tool than a spreadsheet.**

A Question

The first thing that strikes a reader of this work is the odd page numbering. First there is the front matter, then after a divider are Parts A, B, and C. About 60 pages later there is another divider page followed by Part I, which starts on page 3. Why is this not Part D, or Part II since it follows other material?

The answer to that question points out what I see as the book's major weakness: Instead of truly updating the earlier *Executive Computing*, it was left intact and new material was grafted onto it. The first 60 pages or so are new and the following 250 pages are identical to what was published three years before, page numbers and all. That's why page 3 is found some 60 pages into the package.

This decision has a telling effect on the quality of the package. Much of the material in the first 60 pages renders much of the material in the next 250 pages obsolete, particularly to a VisiCalc user.

The VisiCalc section

Parts A, B, and C are new and highlight the VisiCalc program.

Part A contains information on using the diskettes that come with the package. One diskette contains all the VisiCalc templates and the other contains all the BASIC programs that are discussed in the book.

Part B is an introduction to the VisiCalc program for first-time users, but doesn't give enough information about the program to allow a newcomer to use, appreciate, or work with it in analyzing business problems. The newcomer to the VisiCalc program would be hard pressed to modify the upcoming VisiCalc templates into a form that would be helpful professionally. Since the newcomer would still have to go elsewhere for early guidance to the program, and experienced users would learn nothing, little is gained by the introduction.

Part C consists of VisiCalc templates that correspond to business problems discussed later and that were all in the original *Executive Computing*. Not all the problems in the earlier work are handled here with VisiCalc—for good reason, as we'll see.

The templates are relatively simple and easy to follow. For example, one model can be used to calculate an appropriate price for an item to maintain a near constant margin in inflationary times. Another can be used to help decide which are the most profitable products in a product mix.

An experienced VisiCalc user is unlikely to learn much about the program from the models. There is, however, one interesting application that may not have occurred to most users. One template roughs out a corporate cash flow. It contains a loop: as cash-in-hand goes up, debt goes down; as debt goes down, interest goes down; as interest goes down, cash goes up, which further reduces debt, and so on. This loop necessitates a circular reference. Ordinarily when a model with a circular reference is loaded, it comes up with a lot of @ERRORs, which would never resolve.

This problem was handled nicely by putting in a clause for the first cell in the loop that says, essentially, that if I am @ERROR make me a particular value, otherwise calculate me from a formula. The expression in the model is:

```
>D17:@IF(@ISERROR(D17),3637,D28*A17)
```

Here D17 (the interest) depends on D28, which depends on D26, which depends on D23, which depends on D21, which depends on D19, which in turn depends on D17, creating the loop. Since values are set to @ERROR when a sheet is reloaded, if the expression at D17 were "+D28*A17", D17 would come up @ERROR when the model is loaded because D28 ultimately depends on D17 and would also be @ERROR.

By including the @ISERROR clause, D17 is evaluated from @ERROR to 3637, allowing values along the chain to be calculated, from D19 to D21 to D23 and on back to D17. A number of recalculations then will find the correct final values for the cells in the loop. This is a nice way to include circular references in your model if they are necessary. (Notice, however, that not all circular references will resolve.)

Several of the models are hard to grasp until you read the cases to which they apply, found several chapters later. This results from the way the package was put together, with most of the material attached to the front of the pack. It is awkward having to jump between different sections of the book.

At the end of Part C are eight *utilities*, expressions that handle a particular, well-defined problem and are meant to be incorporated into other templates. They correspond to subroutines in languages like BASIC (in fact they are written as subroutines later in this work when BASIC is discussed). Most of the VisiCalc utilities are very simple; one, for example, only sums a series and calculates percentages of a total. There is, however, a useful amortization schedule and a sheet for calculating a stock's systematic risk (beta).

All in all, the VisiCalc section of the package is disappointing. There are few innovative or instructive uses of the VisiCalc program. Its prime virtue is in showing where the VisiCalc program makes problem solving much easier than other computer tools, and this only becomes apparent as one goes through the next 250 pages and refers back.

Another Business Tool

Parts I and II, which follow Parts A, B, and C, are identical to the earlier *Executive Computing* and highlight the BASIC programming language.

The fact that this section makes no reference to the VisiCalc program, yet follows 60 pages devoted to the spreadsheet, makes some passages awkward. For example, the first BASIC program outlines a way to get a fix on pricing goods in inflationary times to keep margins relatively constant. However, a VisiCalc user is unlikely to use such a program since the VisiCalc template described earlier does it so much more easily and is more flexible. This is true with many of the BASIC programs.

I would rather have seen the original book reworked, presenting each problem and then the solutions using BASIC and the VisiCalc program, discussing the relative merits to each approach. Sometimes the VisiCalc program is the better tool; sometimes BASIC or some other programming language is the better tool. For that's all they are, tools, each with strengths and weaknesses. That kind of information can be found in *The Executive Package*, but only implicitly—never are the two tools directly compared. Several of the cases, which took some work in BASIC, are straightforward using the VisiCalc program.

BASIC to the Rescue

On the other hand, there are a number of business problems that a language like BASIC can handle much more efficiently than the VisiCalc program (if the spreadsheet can handle them at all). To a VisiCalc user, the best feature of the *Executive Package* lies in the presentation of such problems.

One interesting example is the problem mentioned above, containing the loop wherein as cash goes up, debt goes down; as debt goes down, interest goes down; as interest goes down, cash goes up, and so on. Using the VisiCalc program, this necessitated a non-intuitive solution. In BASIC the loop is handled naturally as a FOR-NEXT loop. BASIC can be told to execute certain operations as many times as necessary. Here, the program was told to execute the loop until the values in successive passes differed by less than .001. This program illustrates one of BASIC's (and other programming languages') nicer features, the ability to handle loops and iterative processes.

Another model uses a random number generator to deal with probabilistic problems, those that involve a number of independent events each with its own probability of occurrence. One example considers forecasting a stock price given four influences: the economy, take-over bids, competition, and government legislation. The exact effects of these four influences are unknown. There is a certain probability, for example, that the economy will be strong, bolstering the stock price, and a certain probability that the economy will be weak, depressing the price. The same is true for the other factors.

The model takes all these probabilities into account, generating the average and standard deviations of 1000 trials under these probabilities, giving a good idea of where stock price might be expected to go. It is a nice model and one that could not be readily duplicated in the VisiCalc program.

In Part II, pages 99 through 245, the models mostly concern scheduling and planning. From page 149 on it is all BASIC; there are no corresponding VisiCalc templates, although there could and should have been. For example, there is a BASIC program that models inventory given demand and reorder period. It can easily be done using the VisiCalc program, but wasn't. That objection aside, Part II contains much of value to the businessperson.

Chapters 8 and 9 contain excellent discussions of linear programming and decision analysis. Linear programming, which essentially concerns minimizing or maximizing variables within certain constraints, is illustrated by the famous "transportation model". There are three warehouses, each with its own inventory, and three stores, each with its own needs. What is the least expensive way to stock the stores given that cost of shipment depends on the particular warehouse and store?

Although the going gets a little rough here because linear programming involves some fancy manipulations, it is well worth the effort. Maximization is a common business goal. With a bit of work these models can be adapted to particular situations, saving a lot of time. Such problems are extremely difficult to handle on a spreadsheet.

The appendices contain a good primer of BASIC commands and programming structure. There are also 16 BASIC utilities that can be made into subroutines and dumped into programs (this is actually easier to do in

BASIC than in the VisiCalc program). Included are routines for sorting data, plotting, doing histograms and pie charts, and calculating depreciation.

In summary, Nevison's package teaches little about the VisiCalc program to the experienced user and is insufficient for the newcomer. The intermediate user has other, more focused books with which to learn the VisiCalc program for business use. On that score it is not to be recommended. There are, however, some excellent applications of BASIC to common business problems for which a spreadsheet simply isn't appropriate. With a little work, you should be able to pick up enough BASIC to use these models and adapt them to your own needs.

Robert Hildebrand is a staff writer for SATN Publications.

RELATED PRODUCTS AND PUBLICATIONS CHECKLIST

The information contained in this column has been garnered from a variety of sources including dealers, press releases, manufacturers, and advertising.

The first criterion for including any article, book, or newsletter in this listing was whether we could find a hard copy in our library. The second was that the item appeared to be useful. We have included only articles that contain information that can work for you, either in the form of models that you can use or reviews of products that complement the VisiCalc® program.

No one at Software Arts has reviewed or examined any of these products to determine whether they perform as claimed or as listed below. An inclusion in this listing does not constitute an endorsement by Software Arts of that product. We also wish to point out that if any of the products listed alter, patch, or copy the VisiCalc program, they could be in violation of your user licensing agreement and/or the copyright.

Since we have not reviewed any of the products, we cannot provide you with any more information than is contained in the list. If you want to learn more or you wish to purchase any of the products listed, contact your dealer first. Although we have identified distributors in some listings, many distributors do not accept orders from individuals for single copies of software or pieces of hardware.

SOFTWARE

Only software that claims to complement the VisiCalc program will be listed.

Agricultural Templates

AgDisk VisiCalc Templates

- Runs on Apples

Financial statements, depreciation, commodity worksheet, personal budget, and more.

Distributed by Harris Technical Systems, Inc., 624 Peach St., P.O. Box 80837, Lincoln, NE 68501.

Suggested retail price: $75.

AgriCalc*

- Runs on Apple II, Apple ///, TRS-80 Model I, TRS-80 Model III, and the IBM PC

Agricultural management worksheets, including cash flow analysis, depreciation schedules, income and expense sheets, profit/loss statements, loan schedules, and balance sheets.

Distributed by McIntosh Software, 2428 1st Ave NE, Cedar Rapids, IA 52402.

Suggested retail price: $125.

AgriCalc*

- Runs on TRS-80 Model I and TRS-80 Model III

Programs for feeder pig production, feeder cattle production, crop land lease, and general crop profit analysis.

Distributed by Advanced Operating Systems, 450 St. John Rd., Michigan City, IN 46360.

* Although both templates have the name "AgriCalc," they are two different products as far as the SATN staff can determine.

Worksheet

- Runs on Apple II or Apple ///

Fourteen farm decision-aid templates to use with VisiCalc: sprayer calculations, cash flow, cash rent, land purchase, and animal purchase analysis.

Distributed by Harvest Computer Systems, 203 West 11th St., Alexandria, IN 46001.

Suggested retail price: $75.

Financial Templates

Stock Pricing Model

- Runs on the Apple II +

Computes target buy and sell stock prices, earnings acceleration, and return on investment using Dow Jones data.

Distributed by Custom Software, 4652 Cass St., Suite A-238, San Diego, CA 92109.

Suggested retail price: $24.

Option Strategy Model

- Runs on Apple II+

Computes return on investment over a price range for six different option strategies.

Distributed by Custom Software, 4652 Cass St., Suite A-238, San Diego, CA 92109.

Suggested retail price: $24.

PEARformance Measurement System

- Runs on Apple II

Time-weighted total rates of return for stock portfolios, portfolio appraisal, capital gains and losses and more.

Distributed by PEAR Systems Corp., 27 Briar Brae Rd., Stamford, CT 06903.

PUBLICATIONS

Articles

Creative Computing, February 1983. "Apple Cart," by Michael Coffey. A review of *The Power of VisiCalc, Volume 1* with instructions on how to balance your checkbook using VisiCalc.

Harvard Business Review, March/April 1983. "Computerized Sales Management," by G. David Hughes. An article on using VisiCalc for sales projections; time management, planning, and implementation; and account management.

Interface Age, November 1982. "Investing with VisiCalc," by Richard Harkness. Using VisiCalc to analyze lump sum investments and investments in which regular payments are made over a period of time.

Regular Columns

Desktop Computing, July 1983. "Visulating," by Barry Bayer. Personal finance applications in the form of stock market tracking and evaluation.

Softalk, January 1983. "Ventures with VisiCalc," by Joe Shelton. Using Booleans and @ISERROR.

Softalk, July 1983. "Ventures with VisiCalc," by Joe Shelton. Tax/budget applications.

Books

Computer Choices, H. Dominic Covvey and Neil Harding McAlister, Addison-Wesley Publishing Co., 1982. Topics discussed include human engineering, privacy, management of in-house micros, and training.

Controlling Financial Performance for Higher Profits, Dennis P. Curtin, Jeffrey R. Alves, and Anne K. Briggs, Curtin & London, Inc., 1983. There are two books by this title for both the Apple and the IBM PC, with a disk available for each. Topics covered include the computer system needed, income statements and balance sheets, customizing templates, and controlling financial performance.

The DIF File, Donald H. Beil, Reston Publishing Co., 1983. Explores the use of DIF with VisiCalc and other software. Topics covered include general DIF information, case studies, and tutorials.

Planning and Budgeting for Higher Profits, Jeffrey R. Alves, Dennis P. Curtin, and Anne K. Briggs, Curtin & London, Inc., 1983. There are two books by this title for both the Apple and the IBM PC, with a disk available for each. Topics covered include cash budgets, pro forma income statements and balance sheets, and troubleshooting.

32 Different Worksheets for the VisiCalc User, T. G. Lewis, Dilithium Press, 1983. For Apple and IBM PC users, these worksheets includes games, business applications, household applications, and statistical analysis. The sheets may be typed into the computer manually, or purchased on a diskette as a book/software package.

Using VisiCalc: Getting Down to Business, Carol Klitznew and Mathew J. Plociak, Jr., John Wiley & Sons, Inc., 1983. A step-by-step tutorial including one hundred and fifty figures.

VisiCalc Advanced Version Worksheets for Business, Van Wolverton, VisiPress, 1983. This book includes such topics as planning, controlling, and analyzing business opportunities and problems, sales forecasting, budgeting, job costing estimates, and cash flow. Also includes such features in VisiCalc as Keystroke Memory™, Attribute command, label functions, and internal rate of return.

The VisiCalc Book, Atari Edition, Reston Publishing Co., 1982. Intended to supplement VisiCalc's documentation and provide Atari users with an overview of VisiCalc's capabilities.

VisiCalc Disk Guide, David A. Wilson, Osborne/McGraw Hill, 1983. With this DiskGuide you can locate special keys, entries, commands, and files at a glance. Examples show the use of specific commands and functions.

VisiCalc for Science and Engineering, Stanley R. Trost and Charles Pomernacki, Sybex, 1983. Offers more than 50 models for scientific and engineering applications. Topics covered include math and statistics; electrical, mechanical, and solar engineering; heating, ventilating, and air conditioning; control systems, and physics.

VisiCalc Made Easy, David M. Castlewitz, Osborne/McGraw Hill, 1983. Exercises to teach basic skills for designing a worksheet.

HAVE YOU MOVED?

If you have, let us know. To keep your issues of SATN coming to you regularly, we need to know your current address. Send a card or letter, indicating both your old address and your new one, to:

SATN Changes
P.O. Box 99
Newton Lower Falls, MA 02162

Remember to allow about 4 weeks for the change to take effect.